**Single Element in a Sorted Array**

✔️ ***Solution - VI (Binary Search)***

This approach uses the fact that since the array is sorted, every duplicate element occur adjacently. We can also see that the length of given array must be odd (Why? All element occur twice while only 1 element occurs once).

Let the search range be [L, R]. Now, when we consider the mid element, the array is split into two equal halves on the left and right sides. Then we have following cases -

1. nums[mid] == nums[mid+1]:
   * **Left Half Length = Right Half Length = Even :** In this case, we can be sure that our answer lies in the right half. How? One element of right half: nums[mid+1] matches with nums[mid]. So, leaving nums[mid+1] aside, we only have odd number of elements to pair up with each other in the right half. This means one element must be such that it cannot be paired with anyone and this is our required answer that occurs only once. So do **L = mid+2** (+2 to skip nums[mid] and nums[mid+1] & keep search space odd for next iteration)
   * **Left Half Length = Right Half Length = Odd :** In this case, we can be sure that our answer lies in the left half. Again, how? Following same reasoning, one element of left half: nums[mid+1] matches with nums[mid]. So, leaving nums[mid+1] aside, the right half consists of even number of elements which can pair up with each other completely. However, left half only consists odd elements and thus one element which cannot be paired with anyone is our required answer. So do **R = mid-1** (-1 to skip nums[mid] & keep search space odd for next iteration)
2. nums[mid] == nums[mid-1]: We can follow similar reasoning based on array lengths as above -
   * **Left Half Length = Right Half Length = Even :** The answer lies in the left half. How? One element of left half: nums[mid-1] matches with nums[mid]. This leaves left half with odd number of elements to pair up with each other and one element that cant be paired with anyone is the required answer. So do **R = mid-2**
   * **Left Half Length = Right Half Length = Odd :** The answer lies in right half. How? One element of left half: nums[mid-1] matches with nums[mid]. This leaves left half with even number of elements which can pair up with each other completely. However, right half only consists odd elements and thus one element which cannot be paired with any other is the required answer. So do **L = mid+1**.
3. If neither of above condition are satisfied, then we can be sure that nums[mid] itself is the required element occurring once (since it doesn’t match with its neighbors). So, we can just return it.

***Time Complexity :*** **O(logN)**, each time we are eliminating half of search space using binary search  
***Space Complexity :*** **O(1)** only constant extra space is being used.

nums = [1,2,2,3,3,4,4,8,8]

1. [L = 0, R = 8] => mid = 4 and nums[mid] == nums[mid-1]

The left half-length is even and 1 element is equal to nums[mid].

This tells us that left half is left with odd elements to pair up with each other

One element will be left out which is our answer. So, search in left half - [0, 2]

We decremented R by 2 to keep remaining search space of odd length so we can repeat same

Process.

2. [L = 0, R = 2] => mid = 1 and nums[mid] == nums[mid+1]

The left half is of odd length and one element can’t be paired with another.

So, our answer exist in left half. So, search in left half - [0, 0]

3. [L = 0, R = 0] => mid = 1 and nums[mid] != nums[mid-1] and nums[mid] != nums[mid+1]

This means nums[mid] is our final answer since it is not equal to either of its neighbors.